eip ESCUELA INTERNACIONAL POSGRADOS

Máster Avanzado de Programación en Python para Hacking, BigData y Machine Learning

Programación Avanzada Python

LECCIÓN 08

Manipulación de datos

ÍNDICE

INTRODUCCIÓN

En este capítulo veremos como trabajar con grandes conjuntos de datos. Para ello primero veremos como realizar operaciones directamente con archivos y en la segunda parte veremos como trabajar con dataframe, un tipo de datos de la librería Pandas, una de las más utilizadas en Python para el análisis de datos.

OBJETIVOS

Al finalizar esta lección serás capaz de:

1. Realizar operaciones con ficheros.
2. Instalar la librería Pandas.
3. Realizar operaciones con dataframes.
4. Trabajar con CSV

MANIPULACIÓN DE DATOS

Operaciones con archivos

Los archivos pueden contener datos enormes, que están en cualquier formato especificado por su extensión. Python maneja fácilmente los archivos con algunas funciones incorporadas.

Los datos pueden ser leídos desde archivos y escritos en archivos. Primero se carga o abre el archivo y a continuación se realizan las operaciones de lectura o escritura, finalmente se cierra el archivo.

Abrir un archivo

Python utiliza una función incorporada open() para abrir archivos. El parámetro de la función es la ubicación o el nombre del archivo. La función devuelve un objeto archivo que puede utilizarse para realizar posteriores operaciones de lectura o escritura.

Sintaxis:

Objeto_archivo = open ("carpeta/nombre_archivo. extensión")

```
file  = open("/content/sample.txt")
```

Abrir un archivo

Hay dos parámetros opcionales más en la función abrir:

- modo - el modo puede ser especificar para abrir un archivo. Puede ser leer, escribir o añadir. Si el archivo se abre en modo lectura sólo se puede leer del archivo. No se puede hacer ninguna otra operación.
- encode - los caracteres del archivo se codifican por ASCII pero depende de las diferentes plataformas. El tipo de codificación para Windows es 'cp1525'. El tipo de codificación para Linux es 'UTF- 8'

```
f = open("test.txt", mode='r', encoding='utf-8')
```

Cerrar un archivo

Después de realizar todas las operaciones, se debe cerrar el archivo. Ello liberará todos los recursos que se utilizó a lo largo del programa. Se hace con la función incorporada, close()

Podemos simplemente cerrarlo con el objeto archivo de la siguiente manera:

```
f = open("test.txt", encoding = 'utf-8')
# perform file operations
f.close()
```

Abriendo fichero con ‘with’

El archivo puede abrirse mediante la sentencia 'with'. De esta manera no necesita cerrar el archivo explícitamente.

```
with open("test.txt", encoding = 'utf-8') as f:
   # perform file operations
```

Escribir en el archivo

Para escribir en el archivo, este debe ser abierto en los modos

- w - modo de escritura
- a - modo añadir
- x - modo exclusivo

Si se escribe con el modo de escritura, los nuevos datos se sobrescribirán y los antiguos se borrarán.

La función write () se utiliza para escribir cadenas o sentencias en el archivo. Veamos un ejemplo

```
with open("/content/sample.txt",'w') as file:
  file.write("hello\n")
  file.write("sample program\n")
  file.write("to file operations\n")
```

Leer de un archivo

El archivo debe ser leído con el modo r y podemos leer las cadenas en el archivo con la función incorporada read().

```
file = open("/content/sample.txt",'r')
print(file.read())
```

```
hello
sample program
to file operations
```

La función reas() tiene un parámetro size, que se utiliza para dar el número de caracteres para la lectura

```
file = open("/content/sample.txt",'r')
print(file.read(4))
```

```
hell
```

tell()

La función tell se utiliza para obtener la posición actual del cursor en un archivo después de leer todo el contenido

```
file = open("/content/sample.txt",'r')
print(file.read())
```

```
hello
sample program
to file operations
```

```
file.tell()
```

```
40
```

seek()

La función seek se utiliza para mover el cursor a un punto deseado. La posición se da como parámetro.

```
file.seek(20)
print(file.read())
```

```
to file operations
```

readline() vs readlines()

La función readline se utiliza para leer la línea completa.

La función readlines lee las líneas restantes como una lista

```
file = open("/content/sample.txt",'r')
file.readline()
```

```
'hello\n'
```

```
file.readline()
```

```
'sample program\n'
```

```
file.readlines()
```

```
['to file operations\n']
```

```
file = open("/content/sample.txt",'r')
file.readlines()
```

```
['hello\n', 'sample program\n', 'to file operations\n']
```

Trabajando con Pandas

Dataframe de Pandas

El Dataframe de Pandas es un dato tabular con ejes bidimensionales (filas y columnas) que están etiquetados y pueden o no tener valores.

Paquete Pandas:

Pandas se puede instalar en el entorno Python fácilmente usando pip as:

```
pip install pandas
```

Una vez instalado se puede importa el paquete pandas:

```
import pandas as pd
```

pandas.Dataframe ()

Parámetros:

- **data** - se pueden dar datos en forma lista, diccionario, lista de diccionario y ndarry
- **index** - si se especifica el índice, se mostrará la tabla con índice
- **column** - si se especifica el nombre de la columna, se obtendrá la columna correspondiente.
- **dtype** - se puede dar un solo dtype. Por defecto es ninguno.
- **copy** - los datos pueden ser copiados de las entradas. Por defecto será falso.

```
import pandas as pd
list = ['hello','python','dataframe']

df = pd.DataFrame(list)
print(df)
```

	0
0	hello
1	python
2	dataframe

Archivos CSV

El formato de fichero de datos más común es csv. Es un archivo de valores separados por comas que se almacena como una lista de datos que se separan con comas y cuando se abre en Excel u hoja de cálculo se convierte en un formato tabular.

El archivo csv se puede abrir con el atributo read_csv:

pandas.read_csv ()

```
import pandas as pd
csv = pd.read_csv('/content/file.csv')
csv
```

	Player	Pos	AB	H	AVG
0	Denard	OF	545	174	0.319
1	Joe	2B	475	138	0.291
2	Buster	C	535	176	0.329
3	Hunter	OF	485	174	0.359
4	Brandon	SS	5332	125	0.235
5	Eduardo	3B	477	122	0.256
6	Brandon	1B	533	127	0.238
7	Jarrett	OF	215	58	0.270
8	Madison	SP	103	21	0.204

Operaciones con dataframes

Indexación booleana

Si queremos extraer datos del Dataframe basados en algunas condiciones, podemos utilizar la indexación booleana. Se puede recuperar las filas de un criterio particular por el método 'loc':

Dataframe.loc[(condición [nombre_columna])]

```
csv.loc[csv["AVG"] < 0.3]
```

	Player	Pos	AB	H	AVG
1	Joe	2B	475	138	0.291
4	Brandon	SS	5332	125	0.235
5	Eduardo	3B	477	122	0.256
6	Brandon	1B	533	127	0.238
7	Jarrett	OF	215	58	0.270
8	Madison	SP	103	21	0.204

Aplicando funciones

Se puede aplicar una función al Dataframe de pandas. La función puede ser incorporada o definida por el usuario. La función puede aplicarse al marco con el método apply (), sus parámetros son:

- Función - la función que se aplica al conjunto de datos.
- Eje - si el eje es 0, la función se aplica a través de las filas y si el eje es 1, se aplica a las columnas

```
data = {'A': [3,2,4,1,3],
        'B': [33,55,22,44,11]}

df = pd.DataFrame(data)
df
```

	A	B
0	3	33
1	2	55
2	4	22
3	1	44
4	3	11

```
df.apply(sum, axis = 0)
```

```
A     13
B    165
dtype: int64
```

Crosstab ()

Es una función que se utiliza para ver los datos desde otra perspectiva. Ayuda a validar las columnas.

	A	B
0	3	33
1	2	55
2	4	22
3	1	44
4	3	11

```
pd.crosstab(df.A,df.B)
```

B	11	22	33	44	55
A					
1	0	0	0	1	0
2	0	0	0	0	1
3	1	0	1	0	0
4	0	1	0	0	0

Combinando dataframes

Las columnas de los dataframes se pueden fusionar utilizando la función merge (). Los parámetros son los dataframes que se van a fusionar y hay un parámetro más 'on', que sirve para especificar la columna común en el Dataframe.

df1

	Name	Place
0	xyz	asdf
1	abc	erty
2	pqr	dfghj
3	zxc	tyuio
4	fgh	edfgt

df2

	Name	Age
0	xyz	23
1	abc	34
2	pqr	18
3	zxc	55
4	fgh	43

```
df3 = pd.merge(df1,df2,on="Name")

df3
```

	Name	Place	Age
0	xyz	asdf	23
1	abc	erty	34
2	pqr	dfghj	18
3	zxc	tyuio	55
4	fgh	edfgt	43

Ordenando dataframes

Un dataframe puede ser ordenado con dos funciones:

- sort_values () - ordena los dataframes por una o más columnas.
- sort_index () - ordena los dataframes por uno o más índices de fila.

```
df3.sort_values("Age")
```

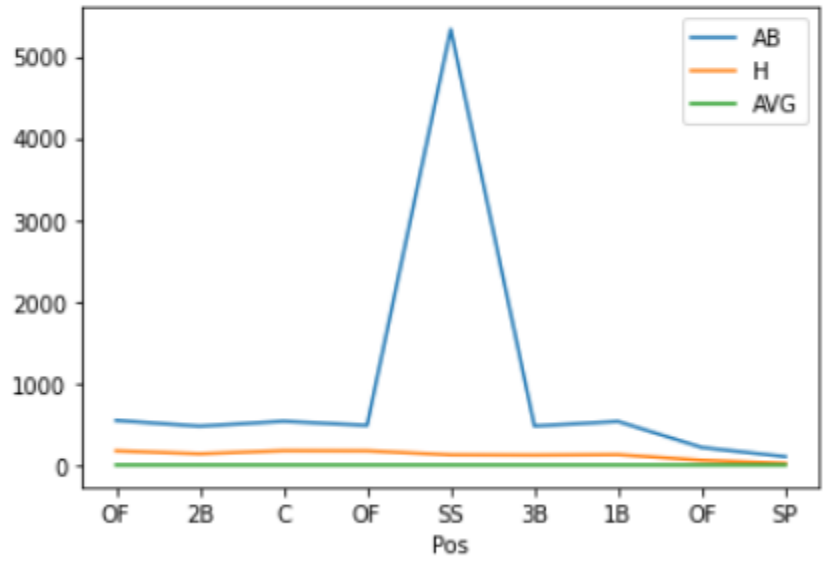
	Name	Place	Age
2	pqr	dfghj	18
0	xyz	asdf	23
1	abc	erty	34
4	fgh	edfgt	43
3	zxc	tyuio	55

Plotting

Los dataframes se pueden dibujar utilizando el método plot (). Cuando llamamos a este método se traza con el índice como eje x y cada columna se traza con diferentes líneas de diferentes colores.

```
csv.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcdd758b990>
```



CONCLUSIONES

1. Python puede leer directamente desde **ficheros**. Esto es muy útil cuando trabajamos con grandes **conjuntos de datos**.

2. **Pandas** es una de las librerías más utilizadas en **Python** para el maneja de grandes **conjuntos de datos**.

3. El **Dataframe** de Pandas es un dato tabular con ejes bidimensionales (filas y columnas) que están etiquetados y pueden o no tener valores.

MUCHAS GRACIAS POR SU ATENCIÓN

eip ESCUELA INTERNACIONAL POSGRADOS

rsanchezi@grupomainjobs.com

Rubén Sánchez Iruela
linkedin.com/in/ruben-sanchez-iruela-8156799a

twitter.com/eiposgrados

facebook.com/eiposgrados

instagram.com/eiposgrados